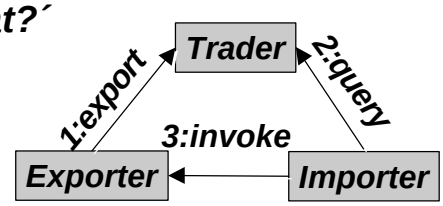
Trading

Motivation

- Locating objects in location transparent way
- Naming simple but may not be suitable when
  - clients do not know server
  - there are multiple servers to choose from
- Trading supports locating servers based on service functionality and quality
- Naming « White pages
- Trading « Yellow Pages

Trading Characteristics

- *Trader operates as broker between client and server.*
- *Enables client to change perspective from ´who?´ to ´what?´*



- *Similar ideas in:*
  - *mortgage broker*
  - *insurance broker*
  - *stock brokerage*

Trading Characteristics

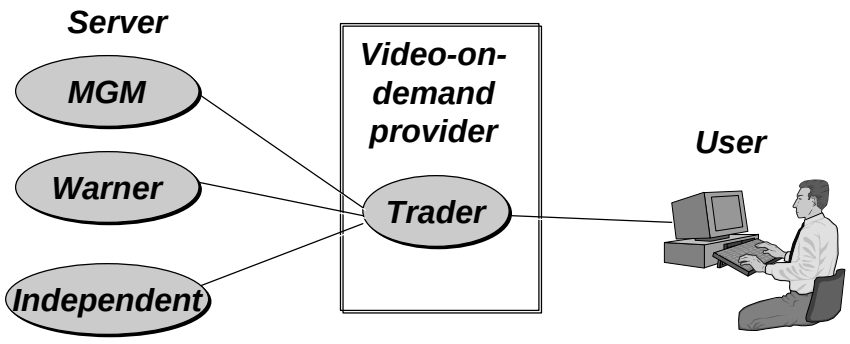
- *Common language between client and server:*
  - *Service types*
  - *Qualities of service*
- *Server registers service with trader.*
- *Server defines assured quality of service:*
  - *Static QoS definition*
  - *Dynamic QoS definition.*

Trading Characteristics

- *Clients ask trader for*
  - *a service of a certain type*
  - *at a certain level of quality*
- *Trader supports*
  - *service matching*
  - *service shopping*

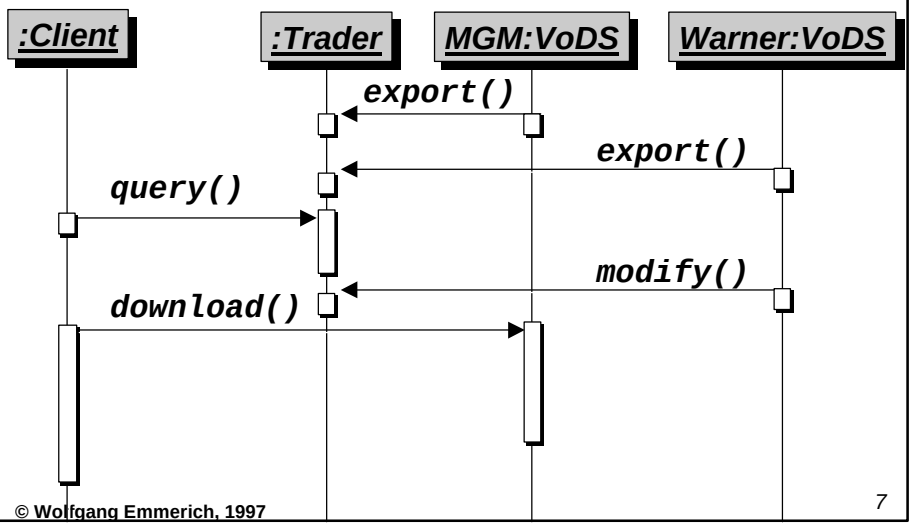
Example

- *Hongkong Telecom video-on-demand server:*



The Trading Process

- *Example: Video-on-demand server*



Service Type Definition

- *Service types define*
  - *Functionality provided by a service and*
  - *Qualities of Service (QoS) provision.*
- *Functionality defined by object type*
- *QoS defined based on properties, i.e.*
  - *property name*
  - *property type*
  - *property value*
  - *property mode*
    - *mandatory/optional*
    - *readonly/modifiable*

Service Type Example

```
typedef enum {VGA,SVGA,XGA} Resolution;
service video_on_demand {
  interface VideoServer;
  readonly mandatory property float fee;
  readonly mandatory property Resolution res;
  optional mandatory property float bandwidth;
}
```


Service Type Hierarchy

- An object type might have several implementations with different QoS.
- Same object type might be used in different service types.
- Service type S is subtype of service S' iff
  - object type of S is identical or subtype of object type of S'
  - S has at least all properties defined for S'
- Subtype relationship can be exploited by trader for service matching purposes

Constraint Definition

- *Importer defines the desired qualities of service as part of the query:*
- *Example:*

```
fee<10 AND res >=SGA AND bandwidth>=256
```

- *In a query, trader matches only those offers that meet the constraint*

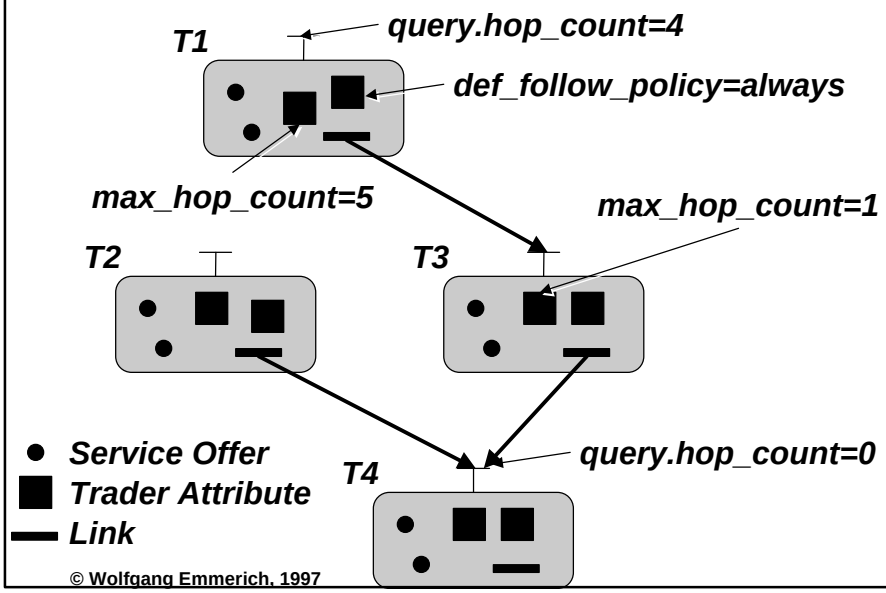
Trading Policies

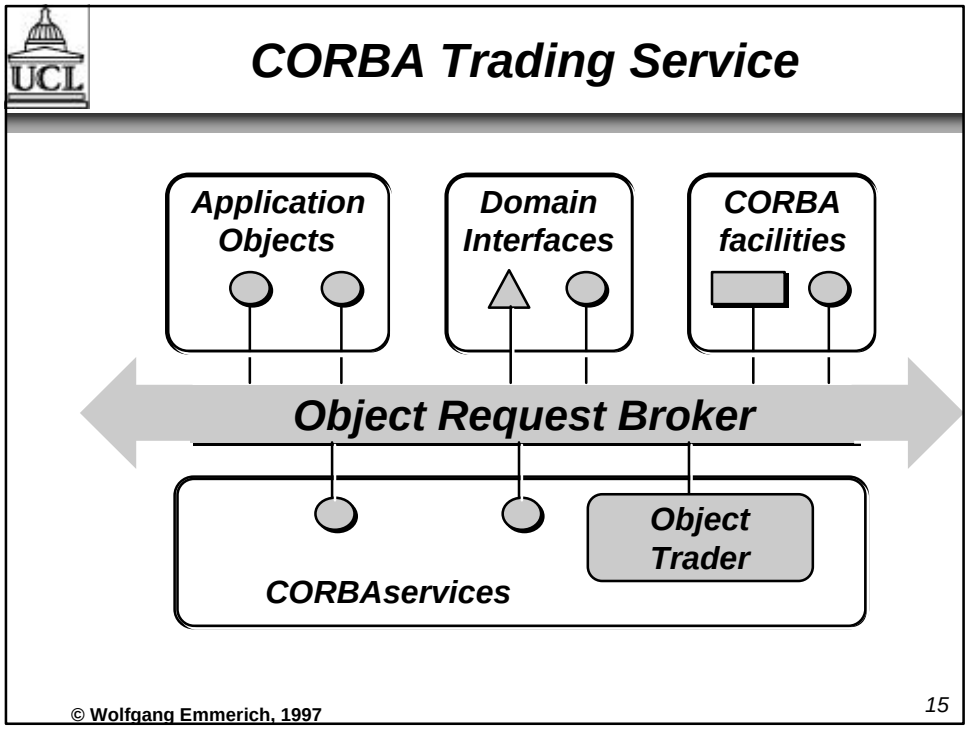
- *Depending on constraint and available services, a large set of offer might be returned by a query.*
- *Trading policies are used to restrict the size of the matched offers*
  - *Specification of an upper limit*
  - *Restriction on service replacements*
  - *Restriction on modifiable properties (these might change between match making and service requests)*

Federated Traders

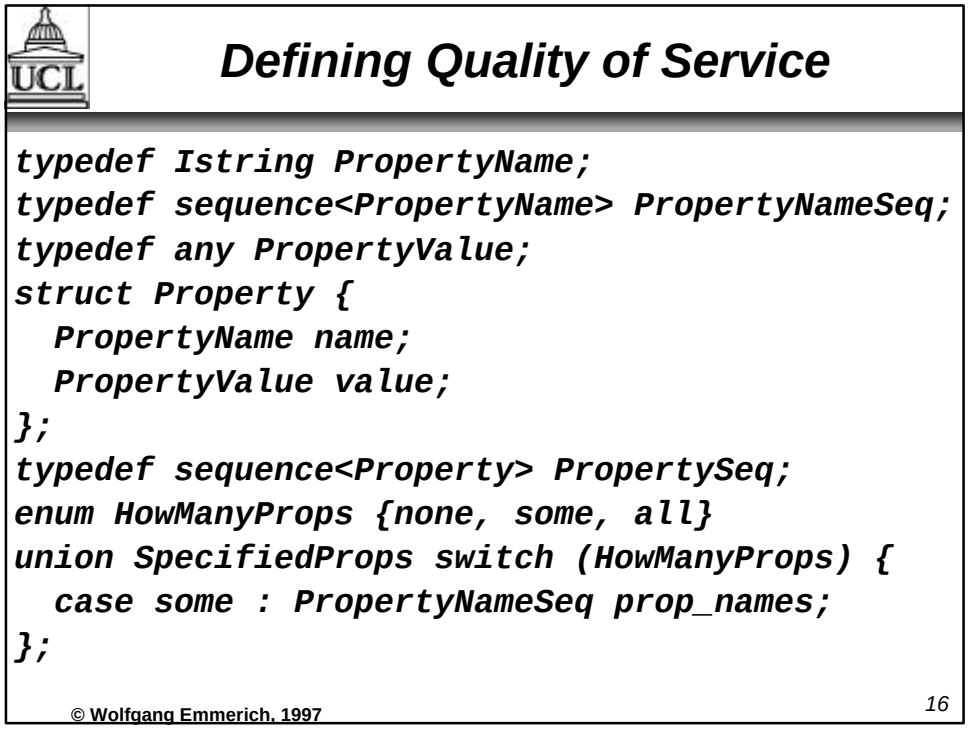
- *Scalability demands federation of traders*
- *A trader participating in a federation*
  - *offers the services it knows about to other traders*
  - *forwards queries it cannot satisfy to other traders*
- *Problems*
  - *Non-termination of import*
  - *Duplication of matched offers*

Trading Graph





CORBA Trading Service



Defining Quality of Service

```
typedef Istring PropertyName;
typedef sequence<PropertyName> PropertyNameSeq;
typedef any PropertyValue;
struct Property {
  PropertyName name;
  PropertyValue value;
};
typedef sequence<Property> PropertySeq;
enum HowManyProps {none, some, all}
union SpecifiedProps switch (HowManyProps) {
  case some : PropertyNameSeq prop_names;
};
```

Trader Interface for Exporters

```
interface Register {
  OfferId export(in Object reference,
                 in ServiceTypeName type,
                 in PropertySeq properties)
                 raises(...);
  OfferId withdraw(in OfferId id)
          raises(...);
  void modify(in OfferId id,
              in PropertyNameSeq del_list,
              in PropertySeq modify_list)
              raises (...);
};
```


Trader Interface for Importers

```
interface Lookup {
  void query(in ServiceTypeName type,
             in Constraint const,
             in Preference pref,
             in PolicySeq policies,
             in SpecifiedProps desired_props,
             in unsigned long how_many,
             out OfferSeq offers,
             out OfferIterator offer_itr,
             out PolicyNameSeq Limits_applied)
             raises (...);
};
```